Batchsize = 128

Lr = 0.001

Epoch = 100

final training accuracy: 0.7806161324050912 final training loss: 0.042796322732771695

final validation accuracy: 0.6391386277604811 final validation loss: 0.11200407892465591

final test accuracy: 0.645120623874979 final test loss: 0.10628158427201785

class Autoencoder\_3\_0(nn.Module):

def \_\_init\_\_(self, dropout\_rate=0.2):

super(Autoencoder\_3\_0, self).\_\_init\_\_()

self.encoder = nn.Sequential(

nn.Conv2d(3, 16, 16, stride=2, padding=1),

nn.ReLU(True),

nn.MaxPool2d(2, stride=2, padding=0),

nn.Conv2d(16, 32, 3, stride=2, padding=1),

nn.ReLU(True),

nn.Dropout(dropout\_rate),

nn.MaxPool2d(2, stride=2, padding=0),

nn.Conv2d(32, 64, 3, stride=2, padding=1),

nn.ReLU(True),

nn.MaxPool2d(2, stride=2, padding=0),

nn.Conv2d(64, 128, 3),

nn.ReLU(True), # Added ReLU here for consistency

nn.MaxPool2d(2, stride=2, padding=0),

)

self.decoder = nn.Sequential(

nn.Upsample(scale\_factor=2, mode='nearest'),

nn.ConvTranspose2d(128, 64, 3),

nn.LeakyReLU(True),

nn.Upsample(scale\_factor=2, mode='nearest'),

nn.ConvTranspose2d(64, 32, 3, stride=2, padding=1, output\_padding=0),

nn.LeakyReLU(True),

nn.Upsample(scale\_factor=2, mode='nearest'),

nn.ConvTranspose2d(32, 16, 3, stride=2, padding=1, output\_padding=1),

nn.LeakyReLU(True),

nn.Dropout(dropout\_rate),

nn.Upsample(scale\_factor=2, mode='nearest'),

nn.ConvTranspose2d(16, 3, 17, stride=2, padding=0, output\_padding=1),

nn.LeakyReLU(True),

nn.Sigmoid(),

)

def forward(self, x):

x = self.encoder(x)

x = self.decoder(x)

return x